# DW_fp_exp

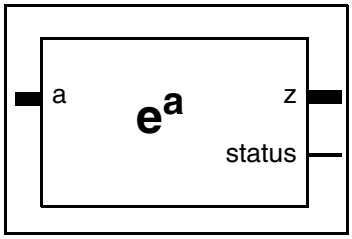
## Floating-Point Exponential ($e^a$)

Version, STAR, and myDesignWare Subscriptions: IP Directory

## Features and Benefits

Revision History

- The precision is controlled by parameters, and covers formats in the IEEE Standard 754
- Exponents can range from 3 to 31 bits
- Fractional part of the floating-point number can range from 2 to 57 bits
- A parameter controls the use of denormal values.




## Description

DW_fp_exp computes the exponential of a floating-point input a, delivering an output $z = e^a$, which is also a floating-point value.

A list of all the parameters used to configure this component is shown in Table 1-2.

The parameter *ieee_compliance* controls the use of denormals and NaNs, as done for other FP operators in the DesignWare Library. When *ieee_compliance* = 0, the operator takes NaN values as infinities, and denormals as zeros. When *ieee_compliance* = 1, the component accepts and generates denormalized values, handles NaN inputs, and delivers NaN outputs when necessary.

Parameters *sig_width* (significand field size) and *exp_width* (exponent field size) define the floating-point format used by input and output operands. Some of these floating-point formats match the formats defined in the IEEE Standard 754.

**Table 1-1 Pin Description**

| Pin Name | Width | Direction | Function |
|---|---|---|---|
| a | (*sig_width* + *exp_width* +1) bits | Input | Input data |
| z | (*sig_width* + *exp_width* +1) bits | Output | Exponential = $e^a$ |
| status | 8 bits | Output | Status flags for the result<br>For details, see STATUS Flags in the *Datapath Floating-Point Overview*. |

**Table 1-2 Parameter Description**

| Parameter | Values | Description |
|---|---|---|
| sig_width | 2 to 57[a] | Word length of fraction field of floating-point numbers a and z |
| exp_width | 3 to 31 | Word length of biased exponent of floating-point numbers a and z |

**Table 1-2 Parameter Description (Continued)**

| Parameter | Values | Description |
|---|---|---|
| ieee_compliance | 0 or 1<br>Default: 0 | Controls the use of denormals and NaNs<br>■ 0 = Do not use denormals or NaNs<br>■ 1 = Use denormals and NaNs |
| arch | 0 to 2<br>Default: 2 | Implementation selection<br>■ 0 = Area optimized<br>■ 1 = Speed optimized<br>■ 2 = Obsolete implementation retained for backward compatibility; will be removed in a subsequent release |

a. The synthesis model fully supports this range, as does the Verilog simulation model in VCS, but the VHDL simulation model (in all simulators) and the Verilog simulation model in non-VCS simulators are limited to a range of 2 - 35.

**Table 1-3 Synthesis Implementations**

| Implementation Name | Function | License Feature Required |
|---|---|---|
| rtl | Implement using the Datapath Generator technology combined with static DesignWare components | DesignWare |

**Table 1-4 Simulation Model**

| Model | Function |
|---|---|
| DW02.DW_FP_EXP_CFG_SIM | Design unit name for VHDL simulation |
| dw/dw02/src/DW_fp_exp_sim.vhd | VHDL simulation model source code |
| dw/sim_ver/DW_fp_exp.v | Verilog simulation model source code |

Given the properties of algorithms to compute the exponential function, and the goal to have a component with good QoR, this component does not have rounding mode control as other FP components in the library. The error is bounded to have a maximum of 2 ulps.

The *arch* parameter controls implementation alternatives for this component. Different values result in different numerical behavior, but the error on the computed values is always bounded by a maximum of 2 ulps. You should experiment with this parameter to find out which value provides the best QoR for your design constraints and technology. Using *arch* = 0 (area optimized implementation) usually provides the best QoR for most time constraints.

For information about the floating-point system defined for the floating-point components, including status flag bits and floating-point formats, refer to the *Datapath Floating-Point Overview*.

## Related Topics

- Datapath Floating-Point Overview
- DesignWare Building Block IP User Guide

## HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp_arith.all;

entity DW_fp_exp_inst is
      generic (
        inst_sig_width : POSITIVE := 10;
        inst_exp_width : POSITIVE := 5;
        inst_ieee_compliance : INTEGER := 0;
        inst_arch : INTEGER := 2
        );
      port (
        inst_a : in std_logic_vector(inst_sig_width+inst_exp_width downto 0);
        z_inst : out std_logic_vector(inst_sig_width+inst_exp_width downto 0);
        status_inst : out std_logic_vector(7 downto 0)
        );
    end DW_fp_exp_inst;

architecture inst of DW_fp_exp_inst is

begin

    -- Instance of DW_fp_exp
    U1 : DW_fp_exp
    generic map (
          sig_width => inst_sig_width,
          exp_width => inst_exp_width,
          ieee_compliance => inst_ieee_compliance,
          arch => inst_arch
          )
    port map (
          a => inst_a,
          z => z_inst,
          status => status_inst
          );

end inst;

-- pragma translate_off
configuration DW_fp_exp_cfg_inst of DW_fp_exp_inst is
 for inst
 end for; -- inst
end DW_fp_exp_cfg_inst;
-- pragma translate_on
```

## HDL Usage Through Component Instantiation - Verilog

```
module DW_fp_exp_inst( inst_a, z_inst, status_inst );

parameter inst_sig_width = 10;
parameter inst_exp_width = 5;
parameter inst_ieee_compliance = 0;
parameter inst_arch = 2;


input [inst_sig_width+inst_exp_width : 0] inst_a;
output [inst_sig_width+inst_exp_width : 0] z_inst;
output [7 : 0] status_inst;

    // Instance of DW_fp_exp
    DW_fp_exp #(inst_sig_width, inst_exp_width, inst_ieee_compliance, inst_arch) U1 (
                .a(inst_a),
                .z(z_inst),
                .status(status_inst) );

endmodule
```

## Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

| Date | Release | Updates |
|---|---|---|
| October 2022 | DWBB_202203.5 | ■ Clarified error bounding in ulps on page 2 |
| March 2019 | DWBB_201903.0 | ■ Clarified value '2' of the arch parameter in Table 1-2 on page 1; this value will be obsoleted in a subsequent release |
| July 2018 | DWBB_201806.1 | ■ For STAR 9001366624, in Table 1-2 on page 1, clarified the range of *sig_width* for the VHDL simulation model (in all simulators) and the Verilog simulation model for non-VCS simulators.<br>■ Added this Revision History table and the document links on this page |

# Copyright Notice and Proprietary Information



### Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

### Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

### Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

### Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

### Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com